UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/674,933	09/30/2003	John Gerard Nistler	ROC920030167US1	5158

7590 11/01/2006

Grant A. Johnson
IBM Corporation , Dept. 917
3605 Highway 52 North
Rochester, MN 55901-7829

EXAMINER	
KIMBALL, MAKAYLA T	

ART UNIT	PAPER NUMBER
2191	

DATE MAILED: 11/01/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/674,933	NISTLER ET AL.	
	Examiner	Art Unit	
	Makayla Kimball	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *30 September 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *30 September 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Claims 1-21 are pending and are considered below.

Claim Objections

1. Claims 7, 8, 13, 16, 18-20 are objected to because of the following informalities:

Claim 7: The letter "s" is missing on the word "exceed"

Claim 8: A colon is missing after "means for calculating" and there should be semicolons where there are commas.

Claim 13: There should be semicolons where there are commas.

Claim 16: There should be an "s" on the word "comprise" and there should be semicolons where there are commas.

Claim 18: There should be an "s" on the word "comprise".

Claims 18-20: There should be a colon after "comprises.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

2. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

3. Claims 14 and 20 recites the limitation "object type". There is insufficient antecedent basis for this limitation in the claim.

4. Claim 21 depends on claim 20 and suffers the same deficiency as claim 20.

Claim Rejections - 35 USC § 101

5. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

6. Claims 11-15 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. The claims appear to be nonstatutory because in the description on page 9, that the signal-bearing medium includes a signal or propagation.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

8. Claims 11 and 13 are rejected under 35 U.S.C. 102(b) as being anticipated by Shaylor (Patent 6,760,907).

Claim 11:

A signal-bearing medium encoded with instructions, wherein the instructions when executed comprises:

Calculating a number of class-type checks; [Column 6, lines 49-54, "...the first case it is known that the class cannot be overridden by subsequent method calls. In this case, method calls can always be called directly or inlined into the code generated for the calling method."]

Generating inline code for the number of class-type checks for a site in a method; and method [Column, 6, lines 55-58, "A still faster optimization is to actually copy the target method's code into the calling method's code (i.e., inlining)."]

Generating an out-of-line function call for any remaining class-type checks at the site that are not handled by the inline code. [As an example, Column 6, lines 53-54, "A direct call is faster to execute than a virtual method call." And Column 7, lines 15-16, "In case four, method calls can be made efficient by calling the method directly."].

Claim 13:

The signal-bearing medium of claim 11, wherein the calculating further comprises:

Calculating the number based on a cost of the inline code; [Column 6, lines 38-40, "The remaining two cases involve gray areas in which some run time knowledge about the class can be exploited to provide generation of more efficient code."]

A cost of the out-of-line function call; and [Column 6, lines 63-67 and Column 7, lines 1-4, "In case two, a simple implementation of the present invention would generate code that implements a traditional virtual function call. This is justifiable because optimization requires greater overhead in such a case and may result in a very large amount of code to be generated for each call to the method. Alternatively, all possible classes are identified as cases in a "switch statement" so that the target method's code from each possible class is inlined into the code for the calling method..."]

A number of times the inline code fails. [Column 6, lines 12-22, "As execution proceeds, profiler 210 stores profile data in the data structure...and indicating whether one class type of a particular class is most likely to be the correct class type at any particular method invocation...it

is dynamically changing to reflect the current behavior of the program in the environment in which it is being used."]

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. The factual inquiries set forth in *Graham* **v.** *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

11. Claims 1-9 and 12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Shaylor (Patent 6,760,907) in view of Tsuboi (Patent 6,848,098).

Claim 1:

Shaylor discloses a method for class-type checks inline [Column 6, lines 49-54, "...the first case it is known that the class cannot be overridden by subsequent method calls. In this case, method calls can always be called directly or inlined into the code generated for the calling method."] to minimize cost at runtime [Column 5, lines 54-56, "The dynamic compiler 208 in accordance with the present invention speeds up the execution of Java code by optimized compilation..."; Column 6, lines 55-57, "A still faster optimization is to actually copy the target

method's code into the calling method's code (i.e., inlining).]. However, Shaylor does not disclose generating a number. In the same field of endeavor, Tsuboi does disclose generating a number [Column 5, lines 39-44, "The optimization unit 7 optimizes or re-optimizes a callable program part so that the optimization level of the callable program part is raised to higher optimization level, when the optimization unit 7 receives the notification of the excess of the number of executions of the callable program part."]. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate generating a number into Shaylor, since Shaylor already discloses generating optimized code. One would have been motivated to incorporate generating a number in order to allow optimization of program code. The optimization of program code enables executing the information processing in a shorter time and reducing memory usage in executing the information processing.

Claim 2:

The method of claim 1, further comprising:

Generating an out-of-line function call for any class-type checks that are not inlined [As an example, Column 6, lines 53-54, "A direct call is faster to execute than a virtual method call." And Column 7, lines 15-16, "In case four, method calls can be made efficient by calling the method directly."].

Claim 3:

The method of claim 1, further comprising:

Calculating the number based on a cost of the inline class-type check [Column 6, lines 38-40, "The remaining two cases involve gray areas in which some run time knowledge about the class can be exploited to provide generation of more efficient code."].

Claim 4:

The method of claim 2, further comprising:

Calculating the number based on a cost of the out-of-line function [Column 6, lines 38-40, "The remaining two cases involve gray areas in which some run time knowledge about the class can be exploited to provide generation of more efficient code."].

Claim 5:

The method of claim 1, further comprising:

Generating a branch hint for a processor if only one class type is encountered at the site [Column 8, lines 52-56, "In contrast, a typical branch instruction can only branch to a location within a limited range of address spaces around the branch instruction. This can be a significant advantage in cases where the patch code is actually compiled at run time."].

Claim 6:

Shaylor discloses an apparatus for class-type checks inline [Column 6, lines 49-54, "...the first case it is known that the class cannot be overridden by subsequent method calls. In this case, method calls can always be called directly or inlined into the code generated for the calling method."] to minimize cost at runtime [Column 5, lines 54-56, "The dynamic compiler 208 in accordance with the present invention speeds up the execution of Java code by optimized compilation..."; Column 6, lines 55-57, "A still faster optimization is to actually copy the target method's code into the calling method's code (i.e., inlining).]. Shaylor also discloses generating inline code for the number of class-type checks for a site in a method [Column, 6, lines 55-58, "A still faster optimization is to actually copy the target method's code into the calling method's

code (i.e., inlining)."]. However, Shaylor does not disclose generating a number. In the same field of endeavor, Tsuboi does disclose generating a number [Column 5, lines 39-44, "The optimization unit 7 optimizes or re-optimizes a callable program part so that the optimization level of the callable program part is raised to higher optimization level, when the optimization unit 7 receives the notification of the excess of the number of executions of the callable program part."]. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate generating a number into Shaylor, since Shaylor already discloses generating optimized code. One would have been motivated to incorporate generating a number in order to allow optimization of program code. The optimization of program code enables executing the information processing in a shorter time and reducing memory usage in executing the information processing.

Claim 7:

Shaylor and Tsuboi disclose the apparatus in claim 6 above. Shaylor discloses generating an -out-of-line function call for any remaining class-type checks at the site [Column 6, lines 63-67 and Column 7, lines 1-4, "In case two, a simple implementation of the present invention would generate code that implements a traditional virtual function call. This is justifiable because optimization requires greater overhead in such a case and may result in a very large amount of code to be generated for each call to the method. Alternatively, all possible classes are identified as cases in a "switch statement" so that the target method's code from each possible class is inlined into the code for the calling method..."]. However, Shaylor does not disclose exceeding a number. In the same field of endeavor, Tsuboi does disclose exceeding a number [Column 5, lines 32-34, "The judging unit 6 determines whether or not the number of executions of a callable program part exceeds a predetermined threshold."]. Therefore, it would have been

obvious to a person of ordinary skill in the art at the time the invention was made to incorporate exceeding a threshold into Shaylor, since Shaylor already discloses generating optimized code. One would have been motivated to incorporate exceeding a threshold to prevent the need of a large memory capacity and to avoid page faults.

Claim 8:

The apparatus of claim 6, wherein the means for calculating the number further comprises:

Means for calculating:

The number based on a cost of the inline code; [Column 6, lines 38-40, "The remaining two cases involve gray areas in which some run time knowledge about the class can be exploited to provide generation of more efficient code."]

A cost of an out-of-line class-type check; and [Column 6, lines 63-67 and Column 7, lines 1-4, "In case two, a simple implementation of the present invention would generate code that implements a traditional virtual function call. This is justifiable because optimization requires greater overhead in such a case and may result in a very large amount of code to be generated for each call to the method. Alternatively, all possible classes are identified as cases in a "switch statement" so that the target method's code from each possible class is inlined into the code for the calling method..."]

A number of times the inline code fails. [Column 6, lines 12-22, "As execution proceeds, profiler 210 stores profile data in the data structure...and indicating whether one class type of a particular class is most likely to be the correct class type at any particular method invocation...it is dynamically changing to reflect the current behavior of the program in the environment in which it is being used."]

Claim 9:

Shaylor and Tsuboi disclose the apparatus as in claim 7 above. Shaylor discloses dynamically recompiling [Column 4, lines 7-8, "The present invention generates code at run time using a dynamic compiler that is responsive to information..."; Column 6, lines 20-22, "some known state at startup, it is dynamically changing to reflect the current behavior of the program in the environment in which it is being used."]. Shaylor also discloses a method of out-of-line function calls [Column 6, lines 63-67 and Column 7, lines 1-4, "In case two, a simple implementation of the present invention would generate code that implements a traditional virtual function call. This is justifiable because optimization requires greater overhead in such a case and may result in a very large amount of code to be generated for each call to the method. Alternatively, all possible classes are identified as cases in a "switch statement" so that the target method's code from each possible class is inlined into the code for the calling method..."]. However, Shaylor does not disclose exceeding a threshold. In the same field of endeavor, Tsuboi does disclose exceeding a number [Column 5, lines 32-34, "The judging unit 6 determines whether or not the number of executions of a callable program part exceeds a predetermined threshold."]. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate exceeding a threshold into Shaylor, since Shaylor already discloses generating optimized code. One would have been motivated to incorporate exceeding a threshold to prevent the need of a large memory capacity and to avoid page faults.

Claim 12:

Shaylor discloses the signal-bearing medium as in claim 11 above. Shaylor also discloses dynamically recompiling [Column 4, lines 7-8, "The present invention generates code at run time using a dynamic compiler that is responsive to information..."; Column 6, lines 20-22, "some

known state at startup, it is dynamically changing to reflect the current behavior of the program in the environment in which it is being used."]. Shaylor also discloses a method of out-of-line function calls [Column 6, lines 63-67 and Column 7, lines 1-4, "In case two, a simple implementation of the present invention would generate code that implements a traditional virtual function call. This is justifiable because optimization requires greater overhead in such a case and may result in a very large amount of code to be generated for each call to the method. Alternatively, all possible classes are identified as cases in a "switch statement" so that the target method's code from each possible class is inlined into the code for the calling method..."]. However, Shaylor does not disclose exceeding a threshold. In the same field of endeavor, Tsuboi does disclose exceeding a number [Column 5, lines 32-34, "The judging unit 6 determines whether or not the number of executions of a callable program part exceeds a predetermined threshold."]. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate exceeding a threshold into Shaylor, since Shaylor already discloses generating optimized code. One would have been motivated to incorporate exceeding a threshold to prevent the need of a large memory capacity and to avoid page faults.

12. Claims 10, and 16-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Shaylor (Patent 6,760,907) in view of Tsuboi (Patent 6,848,098) in further in view of Lueh (Patent 6,658,657).

Claim 10:

Shaylor and Tsuboi disclose the apparatus in claim 6 above. Shaylor discloses to sort inline code [Column 6, lines 12-18, "As execution proceeds, profiler 210 stores profile data in the data

structure indicating whether a particular class has been subclasses, and indicating whether one class type of a particular class is most likely to be the correct class type at any particular method invocation. The data stored is thus more useful than a simple "fixed" indication that can be determined directly from the class description."]. However, Shaylor does not explicitly disclose "frequency". However, Lueh does disclose the inlining based on frequency [Column 4, lines 66-67 and Column 5, lines 1-11, "In order to inline using techniques of the invention, it is preferable that profiling information be collected. Profiling information relates to an instance that is "frequently invoked". It will be appreciated that the system designer or user designates the number of times an instance must be invoked to qualify as being "frequently invoked."...Assume that the instance of foo for class "C" is frequently invoked because class "C's" instance is invoked over a 1,000 times. "C's" foo then is inlined."]. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate "frequency" into Shaylor, since Shaylor already discloses optimizing code. One would have been motivated to incorporate frequency into Shaylor because it reduces the number of memory accesses and improves accuracy of inlining a method.

Claim 16:

Shaylor discloses an electronic device for class-type checks inline [Column 6, lines 49-54, "...the first case it is known that the class cannot be overridden by subsequent method calls. In this case, method calls can always be called directly or inlined into the code generated for the calling method."] to minimize cost at runtime [Column 5, lines 54-56, "The dynamic compiler 208 in accordance with the present invention speeds up the execution of Java code by optimized compilation..."; Column 6, lines 55-57, "A still faster optimization is to actually copy the target method's code into the calling method's code (i.e., inlining).]. Shaylor also discloses generating

inline code for the number of class-type checks for a site in a method [Column, 6, lines 55-58, "A still faster optimization is to actually copy the target method's code into the calling method's code (i.e., inlining)."]. In addition, Shaylor discloses generating an out-of-line function call for any remaining class-type checks at the site [Column 6, lines 63-67 and Column 7, lines 1-4, "In case two, a simple implementation of the present invention would generate code that implements a traditional virtual function call. This is justifiable because optimization requires greater overhead in such a case and may result in a very large amount of code to be generated for each call to the method. Alternatively, all possible classes are identified as cases in a "switch statement" so that the target method's code from each possible class is inlined into the code for the calling method..."]. Shaylor also discloses to sort inline code [Column 6, lines 12-18, "As execution proceeds, profiler 210 stores profile data in the data structure indicating whether a particular class has been subclasses, and indicating whether one class type of a particular class is most likely to be the correct class type at any particular method invocation. The data stored is thus more useful than a simple "fixed" indication that can be determined directly from the class description."]. However, Shaylor does not disclose generating a number, exceeding a number nor does Shaylor explicitly disclose "frequency". In the same field of endeavor, Tsuboi does disclose generating a number [Column 5, lines 39-44, "The optimization unit 7 optimizes or re-optimizes a callable program part so that the optimization level of the callable program part is raised to higher optimization level, when the optimization unit 7 receives the notification of the excess of the number of executions of the callable program part."] and exceeding a number [Column 5, lines 32-34, "The judging unit 6 determines whether or not the number of executions of a callable program part exceeds a predetermined threshold."]. However, Lueh does disclose the inlining based on frequency [Column 4, lines 66-67 and Column 5, lines 1-11, "In order to inline using techniques of the invention, it is preferable that profiling information be collected.

Profiling information relates to an instance that is "frequently invoked". It will be appreciated that the system designer or user designates the number of times an instance must be invoked to qualify as being "frequently invoked."...Assume that the instance of foo for class "C" is frequently invoked because class "C's" instance is invoked over a 1,000 times. "C's" foo then is inlined."]. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate generating a number, exceeding a number and frequently invoked into Shaylor, since Shaylor already discloses generating optimized code. One would have been motivated to incorporate all the above mentioned functions because exceeding a threshold to prevent the need of a large memory capacity and to avoid page faults and generating a number in order to allow optimization of program code. The optimization of program code enables executing the information processing in a shorter time and reducing memory usage in executing the information processing. Also incorporating frequency into Shaylor to reduce the number of memory accesses and improve accuracy of inlining a method.

Claim 17:

The electronic device of claim 16, wherein the calculating further comprises:

Calculating the number based on a cost of the inline code. [Column 6, lines 38-40, "The remaining two cases involve gray areas in which some run time knowledge about the class can be exploited to provide generation of more efficient code."]

Claim 18:

Shaylor and Tsuboi and Lueh disclose the electronic device as in claim 16 above. Shaylor discloses dynamically recompiling [Column 4, lines 7-8, "The present invention generates code at run time using a dynamic compiler that is responsive to information..."; Column 6, lines 20-

22, "some known state at startup, it is dynamically changing to reflect the current behavior of the program in the environment in which it is being used."]. Shaylor also discloses a method of out-of-line function calls [Column 6, lines 63-67 and Column 7, lines 1-4, "In case two, a simple implementation of the present invention would generate code that implements a traditional virtual function call. This is justifiable because optimization requires greater overhead in such a case and may result in a very large amount of code to be generated for each call to the method. Alternatively, all possible classes are identified as cases in a "switch statement" so that the target method's code from each possible class is inlined into the code for the calling method..."]. However, Shaylor does not disclose exceeding a threshold. In the same field of endeavor, Tsuboi does disclose exceeding a number [Column 5, lines 32-34, "The judging unit 6 determines whether or not the number of executions of a callable program part exceeds a predetermined threshold."]. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate exceeding a threshold into Shaylor, since Shaylor already discloses generating optimized code. One would have been motivated to incorporate exceeding a threshold to prevent the need of a large memory capacity and to avoid page faults.

Claim 19:

The electronic device of claim 16, wherein the calculating further comprises:

Calculating the number based on a cost of the out-of-line function call; and [Column 6, lines 63-67 and Column 7, lines 1-4, "In case two, a simple implementation of the present invention would generate code that implements a traditional virtual function call. This is justifiable because optimization requires greater overhead in such a case and may result in a very large amount of code to be generated for each call to the method. Alternatively, all possible classes are

identified as cases in a "switch statement" so that the target method's code from each possible class is inlined into the code for the calling method..."]

A number of times the inline code fails. [Column 6, lines 12-22, "As execution proceeds, profiler 210 stores profile data in the data structure...and indicating whether one class type of a particular class is most likely to be the correct class type at any particular method invocation...it is dynamically changing to reflect the current behavior of the program in the environment in which it is being used."]

13. Claims 14, 20 and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Shaylor (Patent 6,760,907).

As per Claim 14, the rejection of claim 11 is incorporated; however, Shaylor does not disclose:

Calculating a number of class-type checks based on a count of object types encountered at runtime.

Official Notice is taken that it is old and well known within the computing art that an object is an instance of a class. When the objects are counted, so will classes automatically be "counted". One will want to avoid doing redundant work. Therefore, one would be motivated because it will consolidate time.

As per Claim 20, the rejection of claim 16 is incorporated; however, Shaylor does not disclose:

Calculating a number based on a count of object types encountered at runtime.

Official Notice is taken that it is old and well known within the computing art that an object is an instance of a class. A count of an object would be the same as a count of a class. One will not

want to avoid doing redundant work. Therefore, one would be motivated because it will consolidate time.

As per Claim 21, the rejection of claim 20 is incorporated; however, Shaylor does not disclose:

Incrementing the count at runtime.

Official Notice is taken that it is old and well known within the computing art that one will want to increment the counter for each count made. Therefore, one would be motivated because it will produce accurate results.

14. Claim 15 is rejected under 35 U.S.C. 103(a) as being unpatentable over Shaylor (Patent 6,760,907) in view of Lueh (Patent 6,658,657).

Claim 15:

Shaylor discloses the signal-bearing medium in claim 11 above. Shaylor also discloses to sort inline code [Column 6, lines 12-18, "As execution proceeds, profiler 210 stores profile data in the data structure indicating whether a particular class has been subclasses, and indicating whether one class type of a particular class is most likely to be the correct class type at any particular method invocation. The data stored is thus more useful than a simple "fixed" indication that can be determined directly from the class description."]. However, Shaylor does not explicitly disclose "frequency". However, Lueh does disclose the inlining based on frequency [Column 4, lines 66-67 and Column 5, lines 1-11, "In order to inline using techniques of the invention, it is preferable that profiling information be collected. Profiling information relates to an instance that is "frequently invoked". It will be appreciated that the system designer or user

designates the number of times an instance must be invoked to qualify as being "frequently invoked."...Assume that the instance of foo for class "C" is frequently invoked because class "C's" instance is invoked over a 1,000 times. "C's" foo then is inlined."]. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate "frequency" into Shaylor, since Shaylor already discloses optimizing code. One would have been motivated to incorporate frequency into Shaylor because it reduces the number of memory accesses and improves accuracy of inlining a method.

Conclusion

15. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Carini (Patent 5,740,443) – discloses inlining based on two cost functions

"A Comparative Study of Static and Profile-Based Heuristics for Inlining" – discloses use of profile data to inline

Lagergren (Patent 6,964,042) – discloses code optimization using size metrics

Boucher (Patent 6,986,130) – discloses partial function inlining

Adl-Tabatabai et al (Patent 7, 080,354) – discloses dynamic type checking

Sokolov (Patent 6,948,156) – discloses type checking in Java

Plummer et al (Patent 5,579,518) – discloses type checking

16. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Makayla Kimball whose telephone number is 571-270-1057. The examiner can normally be reached on Monday - Thursday 8AM - 2PM EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MTK
10/26/2006

Wei Y Zhen
Supervisory Patent Examiner